Scheduling and Airport Taxiway Path Planning under Uncertainty

Jiaoyang Li* and Han Zhang*
University of Southern California, Los Angeles, CA, 90089

Mimi Gong†, Zi Liang†, Weizi Liu†, Zhongyi Tong†, and Liangchen Yi†
Carnegie Mellon University Silicon Valley, Moffett Field, CA, 94025

Robert Morris‡ and Corina Pasareanu§
NASA Ames Research Center MS-269-1, Moffett Field, CA, 94025

Sven Koenig¶
University of Southern California, Los Angeles, CA, 90089

Congestion and uncertainty on the airport surface are major constraints to the available capacity of the air transport system. This project is to study the problem of planning and scheduling airport surface movement at large airports. Specifically, we focus on the departure time scheduling and taxiway path planning of multiple aircraft under uncertainty. We also developed a simulation tool that is capable of simulating aircraft movement along the taxiway and possible uncertainty during the movement.

I. Introduction

Airport surface operations present a difficult, large-scale logistics problem with a wide range of sub-problems, including: runway sequencing and scheduling; spot or gate release scheduling; gate allocation and taxi route planning and scheduling. Surface movement planning and scheduling is dynamic, with aircraft continuously entering and leaving the operating space. Furthermore, surface movement is unpredictable and prone to unexpected changes in operating conditions due to external factors such as weather. In general, efficiency and safety are difficult objectives to achieve in practice, due to the challenges posed by the presence of uncertainties, human factors, and competing stakeholder interests.

Fast-time simulation of uncertainty in airport surface operations allows for the testing and analysis of modeling concepts and algorithms for planning and scheduling aircraft movement. This paper extends previous and current work in this area [1] [2] by evaluating the result of incorporating probabilistic models into the planning and scheduling process. Our goal is to compare probabilistic approaches to planning and scheduling with standard heuristic approaches that assume a deterministic world, and as a separate stage address uncertainty through continuous re-planning [3]. The goal is to determine whether incorporating recent advances in probabilistic reasoning in planning and scheduling yields more robust schedules through the anticipation of surface delays.

In the remainder of this paper we will provide an overview of a fast-time simulation and scheduling tool for evaluating approaches to planning surface operations, including the technical approach to modeling the geometry and dynamics of airport movement, and the approach to scheduling. We conclude with a summary of the current and expected results of this work for the final draft of the paper.

II. System Overview

The Airport Surface Planner and Simulation System is summarized in Figure 1. The inputs to the main system component, the Simulator, consists of the following:

*PhD Student, Computer Science Department.
†Master Students, Information Networking Institute.
‡Senior Researcher, Intelligent Systems Division.
§Researcher, Intelligent Systems Division/Associate Research Professor, Carnegie Mellon University Silicon Valley.
¶Professor, Computer Science Department.

- An airport node-link model automatically generated from a Google Map page;
- An aircraft dynamics model, for specifying speeds along the taxiway for different types of aircraft;
- A set of shortest path routes to and from gates and runways, used by the scheduler to generate itineraries for each aircraft on the surface;
- A scenario, a problem instance to the scheduler, consisting of a set of arrivals and departure information, including scheduled arrival or departure times, as well as gate and runway information;
- A set of clock parameters to guide the progression of the simulation;
- A set of parameters that describe the level of uncertainty in the problem. These parameters allow for the testing of different levels of uncertainty during simulation;
- A scheduler used in the simulation; and
- A set of test parameters that allow for the collection of statistics when the simulator is in batch mode (i.e. when multiple simulations are run).

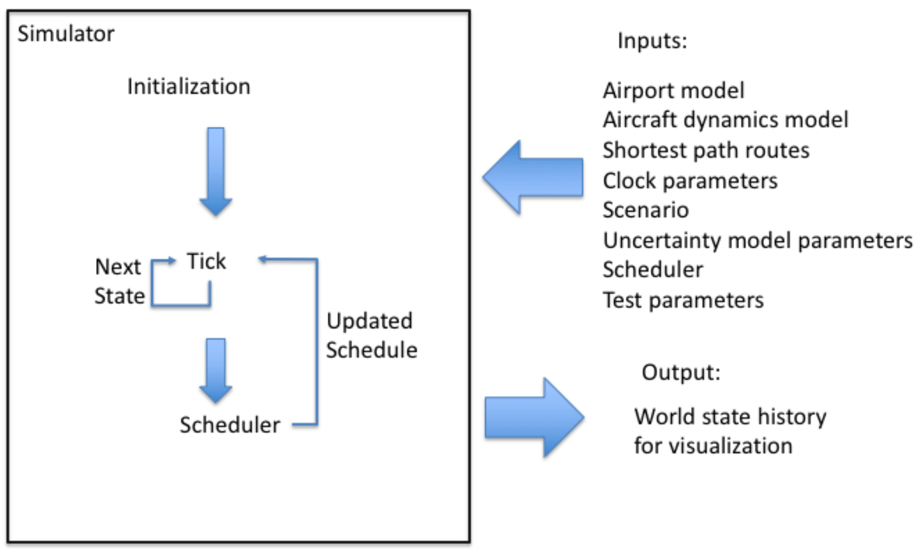

Fig. 1 Airport Surface Planner and Simulation Architecture

The Simulator is comprised of three main components: the initialization, the *Tick* module, and the Scheduler. The Initialization component loads the inputs and models, and invokes the main loop, called the Tick module. The Tick module advances a world state in time, where a world state is comprised of the locations and speeds of all the aircraft. The state also consists of a set of itineraries (schedules) for each aircraft, generated from previous calls to the Scheduler. Tick uses the current state as well as the input data, models and parameters to add, delete or advance aircraft along the airport surface. The output of each run of Tick is a new state. The new state is either fed back directly into the Tick, or, as designated intervals, into the Scheduler. Finally, the Scheduler updates the itineraries associated with the airport state. Specifically, an itinerary is a set of aircraft and associated *targets*, where the targets are nodes on the node-link graph of the airport. The targets are either generated by the pre-compiled shortest path routes, or incrementally inside the scheduler itself. The output of the scheduler is an updated set of conflict free itineraries, which are fed back into the next iteration of the Tick. An itinerary is said to be conflict-free if no pair of aircraft are ever in an unsafe state (violating separation constraints) as the result of executing their associated itineraries simultaneously.

The entire system was implemented in Python and C++. The remainder of this section will explore the main system components in more detail.

A. Node-Link Model and Scenario Generation

The system allows for a node-link model to be generated for any airport from Google Maps. The steps required to build an airport model consist of, first, editing a Google map by adding the desired nodes and edges for runways, taxiways, spots (nodes linking ramp areas with taxiways), and gates; second, downloading the KML file from Google Maps; and third, build a graphical node-link model by running a Python script using the downloaded KML file. It should be noted that the system automatically creates a node-link model with two degrees of resolution: a coarse model that can be used by the Scheduler to generate itineraries; and a fine-grained model that allows for a more realistic simulation and visualization of airport movement.

For faster processing of large airport models, we performed some optimizations. In particular, the routing and scheduling requires a model with nodes for every intersection on the surface, in order to test for conflicts, as discussed below. In addition, for high resolution modeling of movement along a curved link, we split a link into a set of segments that were approximately joined by a straight line. Adding gates and runway terminal nodes results in a graph with a potentially large number of nodes. On the other hand, airports tend to be graphically sparse: each node typically has only one or two nodes coming in or out of it, and relatively few nodes are intersecting nodes. Therefore as a pre-processing step, for the purpose of fast routing, a set of optimizations was performed to limit the graph used for routing to nodes that formed intersections between links.

A random scenario can be generated from a node-link model. The scenario generator is not at this point fully automatic; rather the user must specify a set of arrival and departure *flight templates* using the nodes of the model (gates, runways) and a list of aircraft. The generator will randomly assign arrival or departure times to the templates. The user can set parameters that indicate the 'tightness' of the scenario, i.e., the spacing of the times.

For this paper, we selected a specific flow pattern or arrivals and departures from San Francisco Airport (SFO). Figure 2 shows a node-link model extracted from Google Maps, containing 100 gates, all taxiways and 4 runways that allow for modeling flows on specific pattern called the Southeast Plan. In this flow, aircraft typically arrive on Runway 19L with Runway 19R used as an alternate arrival runway (red lines running northeast/southwest). Departure runways are 10L and 10R (red lines running northwest/southeast). Also modeled are exits and entrances to taxiways from runways, and taxiways (blue lines). Finally, unrestricted areas contain gates and node/links to spot nodes. The result, although a simplification from the complete model of airport flow, is complex enough to test our concepts of planning under uncertainty.

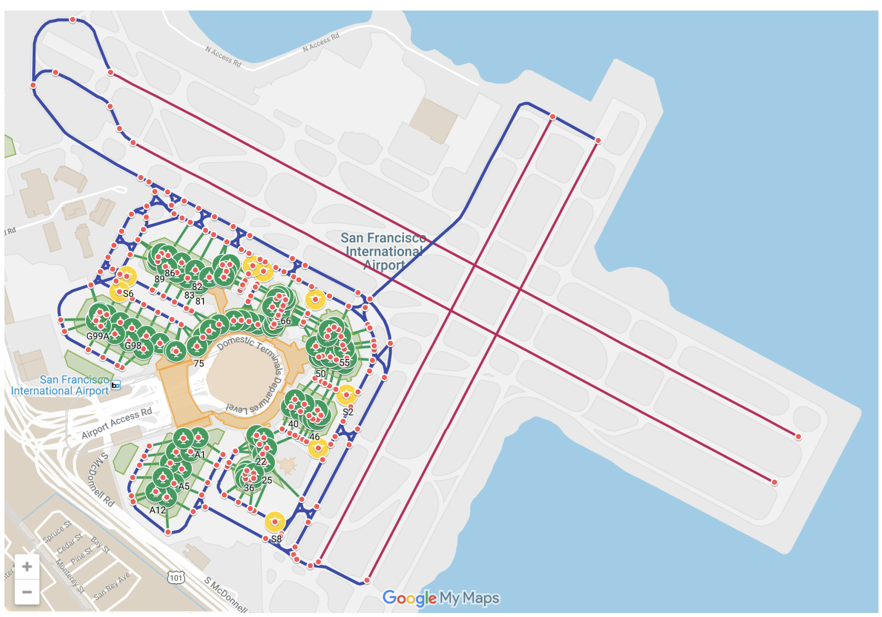

Fig. 2 SFO airport model.

B. Simulator and Visualizer Models

Airport operations is a distributed logistics problem. Consequently, to simulate operations we explicitly model the coordination between a ground controller and pilots for ensuring safe surface movement. Specifically, a Controller module (invokes during a Tick operation) observes the current flow to control the traffic in the intersections. It decides the order of aircraft to pass through each intersection while keeping safe separation between aircraft. The eventual goal of the Controller is to simulate common ATC instructions to pilots, such as "Taxi via...", "Cross Runway..." , "Hold short", etc [4].

An *agent model* is used to simulate aircraft decision making on the surface (this could be the pilot, or an autonomous aircraft towing vehicle, as proposed in [5]. Specifically, this model is used to simulate pilot adjustments to ensure safe distances. In developing the agent model we incorporated recent advances in car-following models for self-driving cars [6].

The *Uncertainty model* allows for the simulation of exogenous events (specifically delays) that affect run-time operations and allows for the testing of more sophisticated scheduling strategies. Delays are injected randomly into the simulator during a Tick operation based on a set of parameters specifying the frequency with which delays occur. Currently, delays can be injected at gate nodes and runway nodes. Injecting a delay during a simulation means that with some probability an aircraft will not advance from its current node (for example, a departure from a gate node) to a target node specified on its itinerary.

The output of the simulator is a sequence of states that provides a complete run of the scenario. This output can be either stored and played back, or streamed in 'real time' by the Visualizer. A window in the display shows the set of active aircraft and their progress along the surface. Figure 3 shows a snapshot of the visualizer replaying a scenario. We plan to add a capability to allow uncertainties to be manually injected during Visualization to allow more fine-tuned testing of scheduling algorithms.

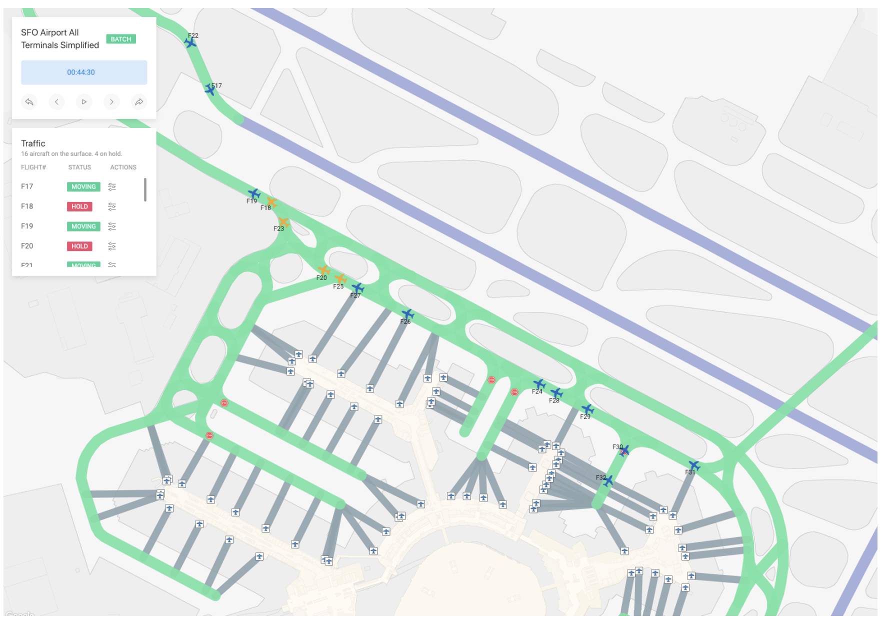

Fig. 3 The Visualizer playing back a scenario.

C. Scheduling

As noted above, the Scheduler is called by the Simulator periodically (a period specified by an input parameter; to model real operations, we chose 15 minutes in world time). The inputs are the current state of the world and the scenario. It outputs a set of paths and departure times (the time to leave the gate nodes for departures), called *itineraries*, one for each active aircraft on the surface.

The algorithm is motivated by the idea that a model of uncertainty can be employed during scheduling to predict delays and so generate schedules that are more robust to unexpected events in the world. Specifically, the scheduling problem becomes a multi-agent path finding problem under uncertainty [7] and use a prioritized planning framework.

The travel time of an aircraft along a link is a random variable which is decided by the previous movement of this aircraft, the travel time of the aircraft that is in front of this aircraft, and the uncertainties of the environment. This is formulated as a probability propagation Markov Chain model which propagates travel time distribution from the aircraft with the highest priority to the aircraft with the lowest priority. We propose two prioritized algorithms: First-Come-First-Serve algorithm (FCFS) where agents with earlier release times have higher priorities and First-Leave-First-Serve algorithm (FLFS) where agents with earlier desired finish times have higher priorities. Once we have the priorities, we use a modified Cooperative A* search [8] to plan paths. In particular, the state in the path-planning search space is specified by a node and a time distribution, and the solution is evaluated by wait times at the gates and the expected travel times.

III. Experimental Results

We need the following experiments:

1) Efficiency/scalability of the scheduler: report the runtime of the scheduler with different numbers of agents (without rolling horizon);
2) Accuracy of the scheduler: compare the finish time from the scheduler with the finish time from the simulator under different uncertainties and different agent densities (without rolling horizon). Note the that under each uncertainty, we need to run the simulator multiple times.
3) Effectiveness and Robustness of the system: compare the solution quality (i.e., wait time and travel time) from the simulator and the workload of the controller (number of stop commands) by using different schedulers (i.e., the baseline algorithm where each aircraft leaves the gate as soon as possible and follows the short path, the two deterministic algorithms that do not consider uncertainties, and the two proposed algorithms) with different uncertainties and different agent densities.

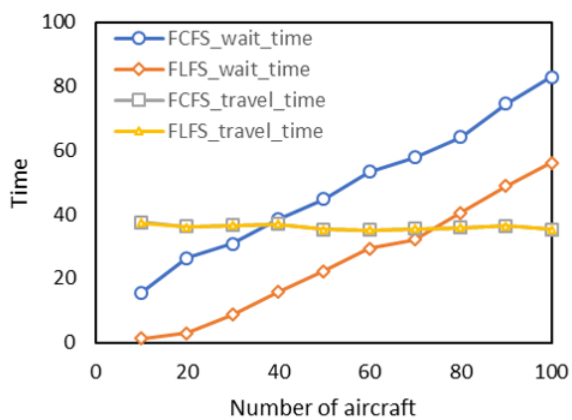

Fig. 4 Solution qualities of FCFS and FLFS.

We compared the two algorithms of Planner/Scheduler, FCFS and FLFS, on a simple node-link model which has 10 gate nodes, 3 spot nodes, 3 taxiway intersection nodes and 1 runway node. Currently we only tested departures. The number of aircraft varies from 10 to 100. For each number of aircraft, we tested 10 random Scenarios and reported the average results. Our code is written in C++, and our experiments are conducted on a 2.80 GHz Intel core i7-7700 laptop with 8 GB RAM.

Figure 4 shows the solution qualities of both algorithms. "wait_time" represents the expected wait time at gates before pushback, and "travel_time" represents the expected travel time from the gate to the runway after pushback. The travel times of both algorithms are similar and stable, which indicates that both algorithms control the taxiway congestion very well. The wait times of both algorithms increase as the number of agents increases, and FLFS has much smaller wait times than FCFS. This is because FLFS first predicts desired finish times of aircraft and assign priorities accordingly. Therefore, FLFS is more effective than FCFS.

Table 1 shows the efficiency of both algorithms. Both algorithms take less than 0.1 s to solve the problem. Although FLFS plans paths twice and FCFS plans paths only once, FLFS still runs faster than FCFS. This is because FCFS usually generates longer paths than FLFS and longer paths needs much more search efforts than shorter paths. Therefore,

Table 1 Efficiency of FCFS and FLFS.

	runtime (ms)		expanded search nodes		generated search nodes	
Aircraft	FCFS	FLFS	FCFS	FLFS	FCFS	FLFS
10	2	1	385	217	595	323
20	6	2	1,058	505	1,697	753
30	10	4	1,741	1,050	2,837	1,600
40	14	8	2,653	1,796	4,413	2,813
50	21	13	3,624	2,679	6,143	4,280
60	28	19	4,899	3,691	8,437	6,040
70	34	24	6,051	4,545	10,487	7,493
80	42	29	7,433	5,947	13,007	9,977
90	53	39	9,325	7,486	16,529	12,786
100	67	48	11,222	9,103	20,074	15,710

FLFS is more efficient than FCFS.

IV. Related Work

This work intersects previous efforts in at least three areas. First, the problem to be solved requires the coordinated planning and scheduling for multiple agents [9]. Second, this work expands upon work on so-called 'rolling horizon' approaches to solve complex scheduling problems under uncertainty [10]. Finally, this work contributes to recent work at building models of uncertainty to improve the robustness of solutions to planning and scheduling problems (for example [7]).

V. Summary

The full paper will describe in more detail a simulation framework that allows for the quantification of the effects of uncertainty on the robustness of solutions generated by surface movement planners and schedulers. We are currently completing the implementation of the agent model and controller, and the full integration of the Simulator with the Probabilistic planning framework. The full paper will consist of experiments using the full system on the complete SFO model. The overall goal of this effort is to investigate and quantify the effects of using recent advances in probabilistic approaches to planning and scheduling to anticipate delays in order to produce more robust plans.

Acknowledgments

References

[1] Montoya, J. V., Windhorst, R. D., Stroiney, S., Griffin, K., Saraf, A., Zhu, Z., and Gridnev, S., "Analysis of Airport Surface Schedulers Using Fast-time Simulation," *AIAA Aviation Technology, Integration, and Operations (ATIO) Conference*, AIAA, Los Angeles, CA, 2013, p. 4275.

[2] Yang, H., Morris, R., and Păsăreanu, C. S., "Analysing the Effect of Uncertainty in Airport Surface Operations," *ISSTA/ECOOP Workshop on International Symposium on Software Testing and Analysis*, ACM, New York, NY, 2018, pp. 132–137.

[3] Malik, W., Gupta, G., and Jung, Y., "Spot release planner: Efficient solution for detailed airport surface traffic optimization," *AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, AIAA, Indianapolis, IN, 2012, p. 5498.

[4] "Operations at Towered Airports," *AOPA Air Safety Foundation*, 2008. `https://www.aopa.org/-/media/files/aopa/home/pilot-resources/asi/safety-advisors/sa07.pdf`.

[5] Morris, R., Chang, M. L., Archer, R., II, E. V. C., Thompson, S., Franke, J. L., Garrett, R. C., Malik, W., McGuire, K., and

Hemann, G., “Self-Driving Aircraft Towing Vehicles: A Preliminary Report,” *AAAI Workshop on Artificial Intelligence for Transportation: Advice, Interactivity, and Actor Modeling*, AAAI Press, Austin, TX, 2015.

[6] Saifuzzaman, M., and Zheng, Z., “Incorporating human-factors in car-following models: A review of recent developments and research needs,” *Transportation Research Part C: Emerging Technologies*, Vol. 48, 2014, pp. 379–403.

[7] Wagner, G., and Choset, H., “Path Planning for Multiple Agents under Uncertainty,” *International Conference on Automated Planning and Scheduling (ICAPS)*, AAAI Press, Pittsburgh, PA, 2017, pp. 577–585.

[8] Silver, D., “Cooperative Pathfinding,” *Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE)*, AAAI Press, Marina del Rey, CA, 2005, pp. 117–122.

[9] Ma, H., and Koenig, S., “Optimal Target Assignment and Path Finding for Teams of Agents,” *International Conference on Autonomous Agents & Multiagent Systems (AAMAS)*, ACM, Singapore, 2016, pp. 1144–1152.

[10] Clare, G., and Richards, A. G., “Optimization of taxiway routing and runway scheduling,” *IEEE Transactions on Intelligent Transportation Systems*, Vol. 12, No. 4, 2011, pp. 1000–1013.